

The opinion in support of the decision being entered today was not written for publication and is not binding precedent of the Board.

Paper No. 11

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte WILLIAM B. BUZBEE
and CARL D. BURCH

Appeal No. 1998-1518
Application No. 08/606,113

ON BRIEF

Before THOMAS, KRASS and BARRETT, Administrative Patent Judges.

KRASS, Administrative Patent Judge.

DECISION ON APPEAL

This is a decision on appeal from the final rejection of claims 1-14, all of the pending claims.

The invention is directed to debugging of computer programs. More particularly, a dynamic translation is employed during debugging of an application. Dynamic translation is translation performed at run time. The application resides in the computing system in

which blocks of code within a shared library are utilized by the application. The blocks of code within the shared library are also available to be utilized by other applications in the system. The dynamic translation includes translation of a first block of code used within the shared library to produce a translated block of code. The translated block of code is included within the translated code. Debugging code, e.g., a break instruction, may then be added to the translated code.

In contrast to prior art methods where the operating system must copy all of the shared code which could possibly be called by an application, the invention is said to require only that shared code actually accessed by the application at runtime or by the debugger is actually translated.

Representative independent claim 1 is reproduced as follows:

1. A method for performing debugging on a first application which resides in a computing system in which a first block of code within a shared library is utilized by the first application and is available to be utilized by other applications in the system, the method comprising the steps of:

(a) dynamically translating the first block of code during runtime of the first application, the dynamic translation producing a translated block of code: and,

(b) placing debugging code within the translated block of code.

The examiner relies on the following references:

Sites	5,507,030	Apr. 9, 1996
		(filed Mar. 7, 1991)
Crank et al. (Crank)	5,583,988	Dec. 10, 1996

Appeal No. 1998-1518
Application No. 08/606,113

(filed Mar. 9, 1994)

Appeal No. 1998-1518
Application No. 08/606,113

Hewlett Packard, PA-RISC 1.1 Architecture and Instruction Set Reference Manual, HP Part No. 09740-09939, Third Edition February 1994, pp. 5-138.

Claims 1-14 stand rejected under 35 U.S.C. § 103. As evidence of obviousness, the examiner offers Sites and Crank with regard to claims 1-4, adding Hewlett Packard with regard to claims 5-14.

Reference is made to the brief and answer for the respective positions of appellants and the examiner.

OPINION

It is the examiner's position, with regard to claim 1, that Sites discloses partially translating an original computer program to provide a translated program, executing the translated program with an interpreter to interpret untranslated portions of the original computer program; and automatically generating a flowgraph used for analyzing a translated program to provide information about blocks of instructions in the flowgraph and for generating translated instructions from the blocks of instructions. The examiner further contends that Sites teaches that the information about the location of untranslated instructions in an original program is discovered during execution of a partial translation of the program but that Sites does not explicitly disclose the capability of placing debugging code within the translated code.

The examiner cites Crank for providing runtime checking of a compiled program to detect errors during program runtime having a debugging capability and for including a

source code editor, a compiler, a linker, a debugger and a library. The examiner then concludes that it would have been obvious to modify Sites to employ the debugging capability of Crank for providing runtime checking of a compiled program to detect errors during program runtime. According to the examiner, the skilled artisan would have been led “to use this debugging capability to detect and correct error within the specified source code or block of code or translated code in order to save time and process” [answer-page 6].

For their part, appellants argue that neither Sites nor Crank discloses or suggests any use of dynamic translation (translation at runtime) for any purpose. Appellants contend that Sites performs all translations at compile time, not at run time, and that Crank’s C source code is compiled and linked all before execution time.

Appellants further argue that because debugging information in the instant invention is inserted during dynamic retranslation at run time, it is possible to limit the placement of debugging code only into blocks of code within a shared library that is actually called by the first application. Sites discloses no modification to code within a shared library and Crank provides for a user to specify restrictions for arguments to individual functions in a library but there is no disclosure, in Crank, of the use of a dynamic translation to insert debugging code within code from a shared library.

Since Sites discloses discovering information about the location of untranslated

instructions in an original program during execution of a partial translation of the program, and that information is used later during re-translation of the original program, it appears that Sites does broadly teach a dynamic translation of a first block of code during runtime of an application wherein the dynamic translation produces a translated block of code. However, independent claim 1, as well as all the other claims, requires that the first block of code be “within a shared library.” It is this claim limitation which we do not find disclosed or suggested by either Sites or Crank.

Sites discloses nothing about a shared library and the examiner has pointed to nothing within Sites alleging such a teaching. Crank does discuss a library, but only with regard to providing for a user to specify restrictions for arguments to individual functions in a library. Thus, barring any employment of impermissible hindsight, it is unclear how the claimed limitation of a first block of code within a shared library, which block of code is dynamically translated during runtime of a first application, is reached by any combination of Sites and Crank.

The examiner explains, at page 12 of the answer, that Site teaches the capability of “code representation including literals, registers, symbol table reference [col. 52, line 5 through col. 54 lines 65. This can be realized as library functions.” The examiner’s explanation is obfuscatory and not persuasive since there is no cited teaching, within Sites, that discusses a first block of code within a shared library which is dynamically translated

during runtime, as claimed. The examiner's further reference to the "runtime checking operations for library functions [col. 7 lines 41-42]" of Crank, adds nothing to the deficiency of Sites regarding a showing of a first block of code within a shared library which is dynamically translated during runtime of a first application.

Because an important claim limitation has not been convincingly shown by the examiner to have been taught or suggested by the applied references [we do not find the Hewlett Packard reference, cited for a teaching of breakpoints, to provide for the deficiencies of Sites and Crank], we hold that the examiner has not established a prima facie case of obviousness with regard to the instant claimed subject matter.

The examiner's decision rejecting claims 1-14 under 35 U.S.C. § 103 is reversed.

REVERSED

JAMES D. THOMAS)	
Administrative Patent Judge)	
)	
)	
)	BOARD OF PATENT
ERROL A. KRASS)	APPEALS AND
Administrative Patent Judge)	INTERFERENCES
)	
)	
)	
LEE BARRETT)	
Administrative Patent Judge)	

Appeal No. 1998-1518
Application No. 08/606,113

EAK/dal
Records Manager
Legal Department 20B0
Hewlett-Packard Company
P.O. Box 10301
Palo Alto, CA 94303-0890